AF IZW

800.0053
A01142

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

OIPE IAP95
AUG 09 2006
PATENT & TRADEMARK OFFICE

In re Application of	:	Pitsianis et al.
For	:	Methods and Apparatus for Indirect VLIW Memory Allocation
Serial No.	:	09/886,855
Filed	:	06/21/2001
Group	:	2124
Examiner	:	Rampuria, Satish

I hereby certify that this correspondence is being deposited with the United States Postal Service as first class mail in an envelope addressed to Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450, on the date set forth below:

Signed: Marianna Tortorelli

Name: Marianna Tortorelli

Date: August 7, 2006

Durham, North Carolina
August 7, 2006

MAIL STOP APPEAL BRIEF – PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF APPELLANT'S BRIEF

Dear Sirs:

1. Transmitted herewith is the APPEAL BRIEF in this application with respect to the Notice of Appeal filed on April 5, 2006.

2. The Applicant is other than a small entity.

3. Pursuant to 37 CFR 1.17(f) the fee for filing the Appeal Brief is $500.00.

[x] The Commissioner is hereby authorized to charge any additional fees which may be required or credit any overpayment to Law Offices of Peter H. Priest Deposit Account No. 50-1058.

Respectfully submitted,

Peter H. Priest
Reg. No. 30,210
Priest & Goldstein, PLLC
5015 Southpark Drive, Suite 230
Durham, NC 27713
(919) 806-1600

800.0053
A01142

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE
BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

OIPE IAP95 AUG 09 2006 PATENT & TRADEMARK OFFICE

In re Application of	:	Pitsianis et al.
For	:	Methods and Apparatus for Indirect VLIW Memory Allocation
Serial No.	:	09/886,855
Filed	:	06/21/2001
Group	:	2124
Examiner	:	Rampuria, Satish

Durham, North Carolina
August 7, 2006

MAIL STOP APPEAL BRIEF – PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPELLANTS' BRIEF

Sir:

1. The Real Party In Interest

The real party in interest is the assignee, PTS Corporation.

2. Related Appeals and Interferences

None.

08/10/2006 MWOLDGE1 00000006 501058 09886855
01 FC:1402 500.00 DA

3. Status of the Claims

This is an appeal from the January 9, 2006 final rejection of claims 1-36, all of the pending claims. Claims 1, 10-15, 19 and 28-33 were rejected under 35 U.S.C. § 103(a) as unpatentable over Faraboschi et al. U.S. Patent No. 5,930,508 ("Faraboschi") in view of Roediger et al. U.S. Patent No. 6,305,014 ("Roediger"). Claims 2-9 and 20-27 were rejected under 35 U.S.C. § 103(a) over Faraboschi and Roediger in view of McKinsey et al. U.S. Patent No. 6,675,380 (McKinsey). Claims 16, 17, 34 and 35 were rejected under 35 U.S.C. § 103(a) over McKinsey in view of Faraboschi. Claims 18 and 36 were rejected under 35 U.S.C. § 103(a) over McKinsey in view of Faraboschi and further in view of G.J. Chaitin, *Register Allocution & Spilling Via Graph Coloring,* ACM, 1982 (Chaitin). Pending claims 1-36 are the subject of this appeal.

4. Status of Amendments

The claims stand as last submitted in the Response to Notice of Non-Compliant Amendment filed November 22, 2005. No Amendment After-Final has been filed.

5. Summary of Claimed Subject Matter

Claims 1 and 19

Claim 1 addresses "a computer implemented method of indirect very long instruction word (VLIW) instruction memory (VIM) allocation" described generally at page 2, line 15-page 16, line 15 and shown in Figs. 1-10. The discussion of details of an embodiment of the method of claim 1 begins at page 7, line 33.

The method of claim 1 comprises "identifying a plurality of VLIW instructions in an input source program". Exemplary details of VLIWs generally are found at page 1, lines 14-page 2, line 14; details for LV (Figs. 2A and 2B and XV (Figs. 3A and 3B) instructions are found

at page 8, line 1-page 9, line 6. As described at page 9, line 35-page 10, line 8 and at page line 34-page 11, line 33 (Figs. 4A and 4B), for example, and as more particularly described at page 11, lines 21 and 22, a compiler, for example, identifies the VLIWs in a program by performing an analysis of the program and then determines "a lifetime of each of said VLIW instructions". Page 11, lines 21 and 22, and lines 29 and 30. See also, page 9, line 12-page 10, line 2 for further discussion of "lifetime"; and page 10, lines 3-8 regarding future aspects of computing lifetime.

At least some of the plurality of VLIW instructions are allocated to VIM based on the lifetime of said plurality of VLIW instructions." See, for example, page 10, lines 15-20; and page 10, line 21-page 14, line 34 for specific detailed examples.

The Official Action notes at page 9 that claim 19 is an apparatus claim corresponding to method claim 1 which is addressed above. That analysis is repeated here with it being noted that a compiler as discussed above is one exemplary "apparatus for allocating indirect very long instruction word (VLIW) instruction memory (VIM)," and comprises "means for identifying a plurality of VLIW instructions in an input source program;

means for determining a lifetime of each of said plurality of VLIW instructions, the lifetime of a VLIW instruction including the interval of time between loading the VLIW instruction to VIM and the last time the VLIW instruction is executed; and

means for allocating at least some of the plurality of VLIW instructions to VIM based on the lifetime of said plurality of VLIW instructions" as claimed in claim 19.

Claims 15 and 33

Claim 15 addresses "a computer implemented method of optimizing the execution time of a user program by reducing redundant loads of very long instruction word (VLIW) instruction

memory (VIM)" described generally at page 2, line 15-page 16, line 15 and shown in Figs. 1-10. More particularly, exemplary discussion of the method of claim 15 begins at page 10, line 3 et seq. with particular attention to page 13, line 24-page 16, line 15.

The method of claim 15 comprises the step of "selecting a load VLIW (LV) instruction in a current mode" as described at page 15, lines 11 and 12 and shown in Fig. 10. Further, the LV instruction is placed "in a new node which is closer to a program start node if an execution frequency of the new node is lower than an execution frequency of the current node, and if a maximum number of VIM lines is not exceeded" as described at page 15, line 1-page 16, and shown in Fig. 10.

The Official Action notes at page 9 that claim 33 is an apparatus claim corresponding to method claim 15 which is addressed above. That analysis is repeated here with it being noted that a compiler as discussed above is an exemplary "apparatus for optimizing the execution time of a user program by reducing redundant loads of very long instruction word (VLIW) instruction memory (VIM)" and comprises "means for selecting a load VIM (LV) instruction in a current node; and

means for placing the LV instruction in a new node which is closer to a program start node if an execution frequency of the new node is lower than an execution frequency of the current node, and if a maximum number of VIM lines is not exceeded" as claimed in claim 33.

Claims 16 and 34

Claim 16 addresses a "a computer implemented method to statically determine liveness of indirect very long instruction word (VLIW) instructions" described generally at page 2, line 15-page 16, line 15 and shown in Figs. 1-10. More particularly, exemplary discussion of the method of claim 16 begins at page 10, line 3 et seq.

The method of claim 16 determines a control flow graph which includes nodes representing basic program blocks, and edges connecting the nodes which represent jumps and calls from one block to another block as described at page 10, lines 4-8, for example.

It also determines "a live-in set and a live-out set of VLIW instructions for each node in the control graph to define a VLIW flow graph, a live-in set for a node comprises the VLIW instructions that are used in the node, a live-out set for a node comprises a union of live-in sets of successor nodes, the determining step further including solving VLIW flow equations for the live-in set and the live-out set" as described at page 11, line 21-page 12, line 23, for example.

The Official Action notes at page 16 that claim 34 is an apparatus claim corresponding to method claim 16 which is addressed above. That analysis is repeated here with it being noted that a compiler as discussed above is one exemplary "apparatus for statically determining liveness of indirect very low instruction word (VLIW) instructions" and comprises " means for determining a control flow graph which includes nodes representing basic program blocks containing VLIW instructions, and edges connecting the nodes which represent jumps and calls from one block to another block; and

means for determining a live-in set and a live-out set of VLIW instructions for each node in the control graph to define a VLIW flow graph, a live-in set for a node comprises the VLIW instructions that are used in the node, a live-out set for a node comprises a union of live-in sets of successor nodes, the determining step further including solving VLIW flow equations for the live-in set and the live-out set" as claimed in claim 34.

Claims 18 and 36

Claim 18 addresses "a computer implemented method to statically determine interference between indirect very long instruction word (VLIW) instructions from a control

graph having a plurality of nodes" described generally at page 2, line 15-page 16, line 15 and shown in Figs. 1-10. More particularly, the discussion of the method of claim 18 begins at page 10, line 3, et seq.

The method of claim 18 determines: "live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph" as described at page 11, line 21-page 12, line 13, for example.

An interference graph is determined "from the VLIW flow graph, the interference graph comprising VLIW nodes in which every VLIW node of the interference graph corresponds to one VLIW instruction" as described at page 12, lines 14-18, for example.

An undirected edge "is inserted into the interference graph between two VLIW nodes if the two VLIW instructions belong to a live-out set of the same node of the VLIW flow graph," as described at page 12, lines 18-20, for example.

The VLIW graph nodes are colored such that adjacent VLIW nodes are colored in different colors and each color corresponds to a different VIM line as described at page 12, lines 20-23, for example.

The Official Action notes at page 14 that claim 36 is an apparatus claim corresponding to method claim 18 which is addressed above. That analysis is repeated here with it being noted that a compiler as discussed above is one exemplary "apparatus statically determining interference between indirect very long instruction word (VLIW) instructions from a control graph having a plurality of nodes" and comprises: "means for determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph

means for determining an interference graph from the VLIW flow graph, the interference graph comprising VLIW nodes in which every VLIW node of the interference graph corresponds

to one VLIW instruction;

means for inserting an undirected edge into the interference graph between two VLIW nodes if the two VLIW instructions belong to a live-out set of the same node of the VLIW flow graph; and

means for coloring the VLIW graph nodes such that adjacent VLIW nodes are colored in different colors and each color corresponds to a different VIM line" as claimed in claim 36.

6. Grounds of Rejection to be Reviewed on Appeal

Claims 1, 10-15, 19, and 28-33 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over Faraboschi in view of Roediger. Claims 2-9 and 20-27 stand rejected under 35 U.S.C. §103(a) over Faraboschi and Roediger in view of McKinsey. Claims 18 and 36 were rejected under 35 U.S.C. 103(a) over McKinsey in view of Faraboschi and further in view of Chaitin. Claims 16, 17, 34 and 35 were rejected under 35 U.S.C. 103(a) over McKinsey in view of Faraboschi.

7. Argument

The final rejection under 35 U.S.C. § 103 did not follow M.P.E.P. § 706.02(j) which states:

> After indicating that the rejection is under 35 U.S.C. 103, the Examiner should set forth...the difference or differences in the claim over the applied reference,...the proposed modification of the applied reference(s) necessary to arrive at the claimed subject matter, and ... an explanation why one of ordinary skill in the art at the time the invention was made would have been motivated to make the proposed modification.

As will be illustrated below, the claims of the present invention are not obvious in view of the references relied upon by the Examiner.

The art rejections are not supported by the relied upon art. All of the rejections are based on Faraboschi and one or more additional items. 35 U.S.C. § 103 which governs obviousness

indicates that "differences between the subject matter sought to be patented and the prior art" are to be assessed based upon "the subject matter as a whole". Analyzing the entirety of each claim, the rejections under 35 U.S.C. § 103 are not supported by the relied upon art as addressed further below.

Only after an analysis of the individual references has been made can it then be considered whether it is fair to combine teachings. However, as addressed further below, fairness requires an analysis of failure of others, the lack of recognition of the problem, and must avoid the improper hindsight reconstruction of the present invention. Such an analysis should consider whether the modifications are actually suggested by the references rather than assuming they are obvious. The 35 U.S.C. § 103 rejections made here pick and choose elements from two or more separate references, which do not collectively present any motivation for making the suggested combination. This approach constitutes impermissible hindsight and must be avoided. As required by 35 U.S.C. § 103, claims must be considered as a whole. When so considered, the present claims are not obvious.

As addressed in greater detail below, Faraboschi, Roediger, McKinsey, and Chaitin do not support the Official Action's reading of them and the rejections based thereupon should be reconsidered and withdrawn. Further, the Applicants do not acquiesce in the analysis of Faraboschi, Roediger, McKinsey and Chaitin made by the Official Action and respectfully continue to traverse the Official Action's analysis underlying its rejections.

A. Rejection under 35 U.S.C. § 103(a) over Faraboschi and Roediger

Claims 1, 10-15, 19 and 28-33

Claims 1, 10-15, 19, and 28-33 were rejected under 35 U.S.C. §103(a) based on

Faraboschi in view of Roediger. Faraboschi describes compacting a very long instruction word in a processor by eliminating no operation (NOP) codes from the VLIW instruction. Faraboschi, col. 2, lines 66-67. The NOP codes in a VLIW instruction correspond to operation codes for functional units that are not needed during execution of the VLIW instruction. Faraboschi, col. 1, lines 52-57. To the end of eliminating NOP codes, Faraboschi's approach involves identifying each word of an instruction that does not contain a NOP code, generating a dispersal code for each identified word, generating a delimiter code for each identified word and storing each identified word along with the corresponding dispersal and delimiter code. Faraboschi, col. 3, lines 1-10. Unlike the present invention, Faraboschi addresses the totally different problem of compacting VLIW instructions by eliminating NOP codes from an instruction. By contrast, the present invention addresses techniques for allocating VLIW instructions to VLIW instruction memory (VIM) regardless of whether the VLIW instruction contains a NOP code or not. See page 2, lines 8-10 of the specification, for example, for a discussion of VIM allocation.

The Examiner suggests **"Faraboschi eliminates the NOP instructions to save the storage area which is similar to efficiently allocating VLIW instructions to memory as disclosed in present invention"** as the basis for rejecting Applicants' arguments regarding Faraboschi (emphasis added). However, this analysis is demonstrably wrong. Faraboschi's efficiency is gained by eliminating NOPs. Thus, an instruction with many NOPs will be compacted greatly, one with some will be compacted somewhat, and one without any NOPs will not be contracted. However, according to claim 1, a VLIW without any NOPs will advantageously be allocated based on its "lifetime" and one with many NOPs won't be depending on its lifetime. The two approaches are very different approaches to gaining efficiency so that one does not make the other obvious.

Claim 1 recites "allocating at least some of the plurality of VLIW instructions to VIM **based on the lifetime**." See also claim 19. With respect to claim 15, the Official Action attempts to equate eliminating NOP codes from a VLIW as taught by Faraboshchi with the objective of the present invention of reducing redundant loads of very long instruction word instruction memory. As indicated above, applicants respectfully disagree. NOP codes are used under the assumption that each execution unit should perform an operation on every processor cycle. Faraboschi, col. 1, lines 41-43. Rather than merely eliminating the storage of NOP codes in memory, claim 15 reduces redundant loading of a load VLIW (LV) instruction by "selecting a load VLIW (LV) instruction in a current node; and placing the LV instruction in a new node which is closer to a program start node if an execution frequency of the new node is lower than an execution frequency of the current node, and if a maximum number of VIM lines is not exceeded." Faraboschi's disclosure is silent with respect to selecting and placing of an LV instruction as presently claimed. See also claim 33. Furthermore, the Official Action admits that Faraboschi does not explicitly disclose determining a lifetime of each of said plurality of VLIW instructions and relies on Roediger accordingly.

Roediger does not cure the deficiencies of Faraboschi. Roediger addresses an instruction scheduler in an optimizing compiler by determining the lifetimes of fixed registers. Roediger, col. 1, lines 62-65. During the compilation, symbolic registers are mapped to fixed registers based on the lifetime of the fixed registers. However, Roediger does not address "allocating at least some of the plurality of VLIW instructions to VIM based on the lifetime of said plurality of VLIW instructions" as claimed in claim 1. (emphasis added)

The Examiner states "**Specifically, the rejection points out that the motivation to 'determine the lifetime of plurality of VLIW instructions' would be to provide the faster**

execution of a computer program so that the compiler does not have to determine the life of a instruction during the run time." However, the relied upon art does not teach determination of "the life of a instruction during the run time" so that even if the first part of the analysis were correct, and it is submitted that such a general motivation is insufficient to support the modification in question, the modification itself is not taught and is not obvious from the relied upon art. Thus, there is no legally adequate basis to combine these two references in the manner outlined by the Examiner. Further, Roediger and Faraboschi cannot simply be combined to meet the presently claimed invention. By way of example, the combination of the compression of VLIW instructions containing NOP codes as taught by Faraboschi with the determination of the lifetime of fixed registers as taught by Roediger would not result in allocating VLIW instructions to VIM as presently taught and claimed. Referring to the specification at page 2, lines 17-21, unlike the Faraboschi and Roediger combination, the present invention advantageously addresses an approach in which a small VIM size may be used even when a software application demands a number of VLIWs larger than can be fit into the physical VIM size of a processor.

The Examiner relies upon the disclosure of page 2, line 5-15 of the present specification as the basis for his position that eliminating NOP codes from instructions is similar to efficiently allocating VLIW instructions to VIM as claimed. Applicants respectfully disagree. The Examiner apparently misreads the cited text. At page 2, lines 5 -7, the cited text reads "indirect VLIW not only avoids the explicit storage of non-operational place holders (nop instructions), but also enables code compression by storing more than one non-overlapping VLIW in the same VIM line." (emphasis added) Although the present specification provides a solution to the problem of eliminating the storage of NOP instructions, the present invention additionally addresses the problem of deciding when a VLIW instruction should be loaded into a VIM line.

Specification, page 2, lines 8-10. To this end, the claimed invention addresses this separate problem by "allocating at least some of the plurality of VLIW instructions to VIM based on the lifetime of said plurality of VLIW instructions," as presently claimed in claim 1. See also claim 19 as presently amended. Claims 15 and 33 address this problem by strategically "placing the LV instruction in a new node which is closer to a program start node if an execution frequency of the new node is lower than an execution frequency of the current node, and if a maximum number of VIM lines is not exceeded."

Furthermore, the Examiner's Response also notes that Roediger is relied upon for purportedly suggesting the limitation "determining a lifetime of each of said plurality of VLIW instructions." As discussed above and in the previous response, Roediger merely describes determining the lifetimes of fixed registers. The lifetime of a fixed register, as defined in Roediger at col. 3, line 66 – col. 4, line 6, is "a set of instructions that operate with that register, either by 'defining' the register (i.e., assigning it a value), or 'using' the register (i.e, reading the current value in the register and using it in a computation). The lifetime contains a single defining instruction, and zero or more subsequent instructions that use the value stored by the defining instruction." Unlike the lifetime of a fixed register storing data used by instructions, the present invention defines a lifetime of a VLIW instruction itself. Claims 1 and 19, as presently amended, recite "the lifetime of a VLIW instruction including the interval of time between loading the VLIW instruction to VIM and the last time the VLIW instruction is executed." Roediger does not teach and does not suggest determining the lifetime of a VLIW instruction as presently claimed.

Moreover, the Examiner's Response implies that Applicants' arguments do not address the combination of references. Applicants respectfully disagree. Even if the compression of

VLIW instructions containing NOP codes as taught by Faraboschi were somehow combined with the determination of the lifetime of fixed registers as taught by Roediger, the combination would still fail to address how to allocate VLIW instructions to VIM and would not meet the claims as presently amended.

B. Rejection under 35 U.S.C. § 103(a) over Faraboschi, Roediger and McKinsey

Claims 2-9 and 20-27 were rejected under 35 U.S.C. §103(a) based on Faraboschi and Roediger in view of McKinsey. McKinsey fails to cure the deficiencies of Faraboschi and Roediger. Since claims 2-9 depend from and contain all the limitations of claim 1, as presently amended, claims 2-9 distinguish from the references in the same manner as claim 1. Since claims 20-27 depend from and contain all the limitations of claim 19, as presently amended, claims 20-27 distinguish from the references in the same manner as claim 19.

Furthermore, McKinsey addresses a path speculating instruction scheduler. McKinsey, Abstract. With McKinsey's system, instructions may be executed out of order to improve the performance of a processor. In so doing, McKinsey's system includes creating control flow graphs made up of blocks of instructions. The control flow graphs are utilized to determine whether a block of instructions can be executed in the same machine cycle, or may be executed in different machine cycles depending on the available resources in the processor and data dependencies between instructions. McKinsey, col. 5, lines 29-42. Unlike the present invention, McKinsey's utilization of control graphs does not address allocation of a VLIW instruction to VIM.

Unlike McKinsey, the present invention utilizes a control graph to determine the lifetime of a VLIW instruction. In general, a VLIW instruction has to be loaded with a load instruction into VIM memory, for example, and subsequently executed with an execute instruction. To

address this special behavior of VLIW instructions, the present invention determines three graphs; a control graph, a VLIW flow graph, and an interference graph. Claim 2, for example, clarifies the step of determining the lifetime of a VLIW instruction when it recites "determining a control flow graph for the input source program containing said plurality of VLIW instructions; determining a VLIW flow graph for said plurality of VLIW instructions; and determining a VLIW interference graph." McKinsey does not disclose and does not make obvious determining the lifetime of a VLIW instruction by "determining a control flow graph for the input source program containing said plurality of VLIW instructions; determining a VLIW flow graph for said plurality of VLIW instructions; and determining a VLIW interference graph," as claimed in claim 2. Further, it provides no basis for modifying Faraboschi and Roediger so that the combination meets the present claims.

C. Rejection under 35 U.S.C. § 103(a) over McKinsey in view of Faraboschi and Chaitin

Claims 18 and 36 were rejected under 35 U.S.C. §103(a) based on McKinsey in view of Faraboschi and further in view of Chaitin. Claim 18, as presently amended, recites the step of "determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph." While McKinsey discloses a control graph, also referred to as a dependence graph, for instructions which do not have VLIW instruction behavior as discussed above, McKinsey does not teach and does not suggest defining a VLIW flow graph from a control graph and "determining an interference graph from the VLIW flow graph" as presently claimed in claims 18 and 36. Furthermore, McKinsey does not teach and does not suggest "inserting an undirected edge into the interference graph between two VLIW nodes," as presently claimed in claims 18 and 36. (emphasis added). The only graphs disclosed in

McKinsey require directed edges.

Faraboschi fails to cure the deficiencies of McKinsey. Faraboschi is relied upon for compacting VLIW instruction. As discussed above, the present invention is not merely compacting an individual VLIW instruction by removing a NOP code. Rather, claims 18 and 36 address determining "interference between indirect very long instruction word (VLIW) instructions."

The present invention addresses coloring the VLIW graph nodes of a VLIW graph which is not a data-flow graph. The VLIW graph was obtained from determining the live-out sets of the control graph. Claim 18, as presently amended, recites "determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph." The relied upon art does not teach and does not suggest "coloring the VLIW graph nodes such that adjacent VLIW nodes are colored in different colors and each color corresponds to a different VIM line," as presently claimed in claims 18 and 36.

Even if there was a basis for combining the teachings of Faraboschi, Roediger and McKinsey, such combination fails to meet the elements of the present claims. The combination of the control graph of McKinsey, the compressed VLIW instruction of Faraboschi, does not teach and does not suggest "determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph; determining an interference graph from the VLIW flow graph ... inserting an undirected edge into the interference graph between two VLIW nodes ... [and] coloring the VLIW graph nodes," as presently claimed in claim 18.

D. Rejection under 35 U.S.C. § 103(a) over McKinsey in view of Faraboschi

Claims 16, 17, 34 and 35

Claims 16-17 and 34-35 were rejected under 35 U.S.C. §103 based on McKinsey in view

of Faraboschi. The deficiencies of McKinsey and Faraboschi discussed above apply here as well. Claim 16, as presently amended, recites "determining a live-in set and a live-out set of VLIW instructions for each node in the control graph to define a VLIW flow graph, a live-in set for a node comprises the VLIW instructions that are used in the node, a live-out set for a node comprises a union of live-in sets of successor nodes, the determining step further including solving VLIW flow equations for the live-in set and the live-out set." The Official Action relies on col. 10, lines 7-9 of McKinsey to purportedly suggest determining the liveness of indirect very long instruction word (VLIW) instructions. Applicants respectfully disagree with this reading of McKinsey. At the cited portion of text, the "live on exit" value is computed when a specific instruction requests so and is "used by instructions outside the region R." McKinsey fails to discuss how or under what conditions instructions outside the region R use this "live on exit" value.

Unlike McKinsey and Faraboschi, the present invention determines liveness of a VLIW by determining a control flow graph and determining live-in sets and live-out sets of VLIW instruction for each node in the control graph. McKinsey and Faraboschi, separately or in combination, do not teach and do not suggest "determining a live-in set and a live-out set of VLIW instructions for each node in the control graph to define a VLIW flow graph, a live-in set for a node comprises the VLIW instructions that are used in the node, a live-out set for a node comprises a union of live-in sets of successor nodes, the determining step further including solving VLIW flow equations for the live-in set and the live-out set," as presently claimed in claim 16.

The relied upon references fail to recognize and address the problem of allocating VLIW instruction memory in the manner advantageously addressed by the present claims. The claims

as presently amended are not taught, are not inherent, and are not obvious in light of the art relied upon.

Overall, Applicant is somewhat puzzled by the Examiner's response to the previously submitted arguments and the apparent refusal of the Examiner to consider both the plain language and the context of the present claims. The relied upon references do not teach and do not render obvious the present claims which performs the presently claimed functions.

To sum up, the relied upon art does not show and does not methods and systems for indirect VLIW memory allocation as presently claimed. Nothing in the cited references indicates a recognition of the problems addressed by the present invention. The claims of the present invention are not taught, are not inherent, and are not obvious in light of the art relied upon.

E. The Examiner's Findings of Obviousness are Also Contrary to Law of the Federal Circuit

As shown above, the invention claimed is not suggested by the relied upon prior art. The references cited by the Examiner, if anything, teach away from the present invention. It is only in hindsight, after seeing the claimed invention, that the Examiner could combine the references as the Examiner has done. This approach is improper under the law of the Federal Circuit, which has stated that "[w]hen prior art references require selective combination by the Court to render obvious a subsequent invention, there must be some reason for the combination other than the hindsight gleaned from the invention itself." Uniroyal, Inc. v. Rudkin-Wiley Corp., 837 F.2d 1044, 1051, 5 U.S.P.Q. 2d 1434, 1438 (Fed. Cir. 1988), cert. den., 109 S. Ct. 75, 102 L.Ed. 2d 51 (1988); quoting Interconnect Planning Corp. v. Feil, 774 F.2d 1132, 1132, 227 U.S.P.Q. 543, 535 (Fed. Cir. 1985). Furthermore, "[i]t is impermissible to use the claims as a frame and the prior art references as a mosaic to piece together a facsimile of the claimed invention." Uniroyal, 837 F.2d at 1051, 5 U.S.P.Q. 2d at 1438. Similarly, "[t]he mere fact that the prior art could be so

modified would not have made the modification obvious unless the prior art suggested the desirability of the modification." In re Laskowski, 871 F.2d 115, 117, 10 U.S.P.Q. 2d 1397, 1398 (Fed. Cir. 1989), quoting In re Gordon, 733 F.2d 900, 902, 221 U.S.P.Q. 1125, 1127 (Fed. Cir. 1984). No such suggestion is found here.

In addition, the Examiner does not appear to have considered "where the references diverge and teach away from the claimed invention", Akzo N.V. v. International Trade Commission, 808 F.2d 1471, 1481, 1 U.S.P.Q. 2d 1241, 1246 (Fed. Cir. 1986), cert. den., 107 S. Ct. 2490, 482 U.S. 909, 107 S.Ct. 2490 (1987); and W.L. Gore Associates, Inc., 721 F.2d 1540, 220 U.S.P.Q. 303 (Fed. Cir. 1983); nor has the Examiner read the claims as a whole, as required by statute. 35 U.S.C. §103. See also, Smithkline Diagnostics Inc. v. Helena Laboratories Corp., 859 F.2d 878, 885, 8 U.S.P.Q. 2d 1468, 1475 (Fed. Cir. 1988); and Interconnect Planning Corp., 774 F.2d at 1143, 227 U.S.P.Q. at 551.

In re Laskowski, 871 F.2d 115, 10 U.S.P.Q. 2d 1397, the Federal Circuit reversed an obviousness rejection of the claims in an application for a bandsaw. The claimed bandsaw used a pulley type wheel loosely fitted with a tire. The primary reference showed a similar bandsaw where the band was tightly fitted. The Federal Circuit stated that the prior art did not provide a suggestion, reason or motivation to make the modification of the reference proposed by the Commissioner. Id. at 1398. The Court added that "there must be some logical reason apparent from the positive, concrete evidence of record which justifies a combination of primary and secondary references." Id. quoting In re Regel, 526 F.2d 1399, 1403, 188 U.S.P.Q. 136, 139 (C.C.P.A. 1975), citing In re Stemniski, 444 F.2d 581, 170 U.S.P.Q. 343 (C.C.P.A. 1971).

In Uniroyal Inc. v. Rudkin-Wiley Corp., 837 F.2d 1044, 5 U.S.P.Q. 2d 1434 (Fed. Cir. 1988), cert. den., 109 S. Ct. 75, 102 L.Ed. 2d 51 (1988), the Federal Circuit reversed the District

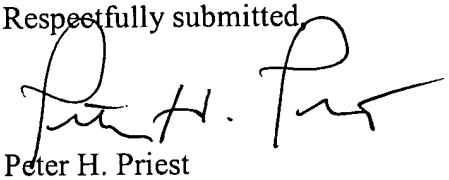
Court's finding that the claims for a patent for an air flow deflecting shield were obvious. Without any suggestion in the art, the District Court improperly chose features from several prior art references to recreate the claimed invention.

The Examiner's rejection suggests that the Examiner did not consider and appreciate the claims as a whole. The claims disclose a unique combination with many features and advantages not shown in the art. It appears that the Examiner has oversimplified the claims and then searched the prior art for the constituent parts. Even with the claims as a guide, however, the Examiner did not recreate the claimed invention.

8. Conclusion

The rejection of claims 1-36 should be reversed and the application promptly allowed.

Respectfully submitted,



Peter H. Priest
Reg. No. 30,210
Priest & Goldstein, PLLC
5015 Southpark Drive, Suite 230
Durham, NC 27713
(919) 806-1600

CLAIMS APPENDIX
(Claims Under Appeal)

1. A computer implemented method of indirect very long instruction word (VLIW) instruction memory (VIM) allocation comprising the steps of:

identifying a plurality of VLIW instructions in an input source program;

determining a lifetime of each of said plurality of VLIW instructions, the lifetime of a VLIW instruction including the interval of time between loading the VLIW instruction to VIM and the last time the VLIW instruction is executed; and

allocating at least some of the plurality of VLIW instructions to VIM based on the lifetime of said plurality of VLIW instructions.

2. The method of claim 1 wherein the step of determining the lifetime of each of said plurality of VLIW instructions further comprises the steps of:

determining a control flow graph for the input source program containing said plurality of VLIW instructions;

determining a VLIW flow graph for said plurality of VLIW instructions; and

determining a VLIW interference graph.

3. The method of claim 2 wherein the step of determining the VLIW flow graph further comprises the step of:

solving VLIW flow equations.

4. The method of claim 2 wherein the control flow graph includes:

a plurality of nodes which correspond to basic blocks of the VLIW instructions; and

a plurality of edges, wherein each edge corresponds to a jump or a call from a given basic block to another basic block.

5. The method of claim 4 wherein the flow control graph at each of said plurality of nodes includes:

at least one VLIW instruction defined by the node; and

at least one VLIW instruction used by the node.

6. The method of claim 5 further comprising the step of:

determining live-in sets and live-out sets for each of said plurality of nodes.

7. The method of claim 6 wherein the VLIW flow graph comprises the control flow graph and the live-in sets and live-out sets for each of said plurality of nodes.

8. The method of claim 7 wherein the step of allocating VIM further includes the step of:

determining an interference graph in which every node of the interference graph corresponds to one of said plurality of VLIW instructions.

9. The method of claim 8 wherein the VIM comprises a plurality of VIM lines, and the step of determining an interference graph further comprises the steps:

inserting an undirected edge into the interference graph between two VLIW nodes if the two VLIW instructions belong to a live-out set of the same node of the VLIW flow graph; and

coloring the interference graph nodes such that adjacent interference nodes are colored in different colors and each color corresponds to a different VIM line.

10. The method of claim 1 wherein the lifetime of a VLIW instruction is a time interval extending from when said VLIW is defined by a load VLIW instruction to when said VLIW is last executed by an execute VLIW instruction.

11. The method of claim 1 further comprising the step of:

shortening the life of a particular VLIW by placing an initialization load VLIW (LV)

statement adjacently prior to the use of its corresponding execute VLIW (XV) statement.

12. The method of claim 1 further comprising the step of:

merging two non-overlapping VLIWs to share a common VIM line only when color ability of a resulting VLIW interference graph does not worsen as a result of said merging.

13. The method of claim 1 further comprising the step of:

utilizing a coalescing heuristic to reduce VIM requirements of a program.

14. The method of claim 13 wherein said step of utilizing a coalescing heuristic results in a coalesced VIM address holding two or more of said plurality of VLIW instructions.

15. A computer implemented method of optimizing the execution time of a user program by reducing redundant loads of very long instruction word (VLIW) instruction memory (VIM) comprising the steps of:

selecting a load VLIW (LV) instruction in a current node; and

placing the LV instruction in a new node which is closer to a program start node if an execution frequency of the new node is lower than an execution frequency of the current node, and if a maximum number of VIM lines is not exceeded.

16. A computer implemented method to statically determine liveness of indirect very long instruction word (VLIW) instructions comprising the steps of:

determining a control flow graph which includes nodes representing basic program blocks, and edges connecting the nodes which represent jumps and calls from one block to another block; and

determining a live-in set and a live-out set of VLIW instructions for each node in the control graph to define a VLIW flow graph, a live-in set for a node comprises the VLIW instructions that are used in the node, a live-out set for a node comprises a union of live-in sets of

successor nodes, the determining step further including solving VLIW flow equations for the live-in set and the live-out set.

17. The method of claim 16 wherein the VLIW flow equations comprise:

$I_n = U_n \cup (O_n - D_{KNY})$; and

$O_n = \cup_{s \text{ in succ}(n)} I_s$;

where "n" is a given node, I_n is a set of live-in VLIWs at node "n", O_n is a set of live-out VLIWs at node "n", U_n is a set of VLIWs that are used in "n", D_n is a set of VLIWs that are defined in "n", the live-out VLIWs of node "n" are all the VLIWs that belong to live-in sets of successor nodes of "n", and the notation $\cup_{s \text{ in succ}(n)} I_s$ denotes the union of all sets I_s where s is a successor node to node n.

18. A computer implemented method to statically determine interference between indirect very long instruction word (VLIW) instructions from a control graph of having a plurality of nodes, the computer implemented method comprising the steps of:

determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph;

determining an interference graph from the VLIW flow graph, the interference graph comprising VLIW nodes in which every VLIW node of the interference graph corresponds to one VLIW instruction;

inserting an undirected edge into the interference graph between two VLIW nodes if the two VLIW instructions belong to a live-out set of the same node of the VLIW flow graph; and

coloring the VLIW graph nodes such that adjacent VLIW nodes are colored in different colors and each color corresponds to a different VIM line.

19. An apparatus for allocating indirect very long instruction word (VLIW)

instruction memory (VIM) comprising:

means for identifying a plurality of VLIW instructions in an input source program;

means for determining a lifetime of each of said plurality of VLIW instructions, the lifetime of a VLIW instruction including the interval of time between loading the VLIW instruction to VIM and the last time the VLIW instruction is executed; and

means for allocating at least some of the plurality of VLIW instructions to VIM based on the lifetime of said plurality of VLIW instructions.

20. The apparatus of claim 19 wherein the means for determining the lifetime of each of said plurality of VLIW instructions further comprises:

means for determining a control flow graph for the input source program containing said plurality of VLIW instructions;

means for determining a VLIW flow graph for said plurality of VLIW instructions; and

means for determining a VLIW interference graph.

21. The apparatus of claim 20 wherein the means for determining the VLIW flow graph further comprises:

means for solving VLIW flow equations.

22. The apparatus of claim 20 wherein the control flow graph includes:

a plurality of nodes which correspond to basic blocks of the VLIW instructions; and

a plurality of edges, wherein each edge corresponds to a jump or a call from a given basic block to another basic block.

23. The apparatus of claim 22 wherein the flow control graph at each of said plurality of nodes includes:

at least one VLIW instruction defined by the node; and

at least one VLIW instruction used by the node.

24. The apparatus of claim 23 further comprising:

means for determining live-in sets and live-out sets for each of said plurality of nodes.

25. The apparatus of claim 24 wherein the VLIW flow graph comprises the control flow graph and the live-in sets and live-out sets for each of said plurality of nodes.

26. The apparatus of claim 25 wherein the means for allocating VIM further includes:

means for determining an interference graph in which every node of the interference graph corresponds to one of said plurality of VLIW instructions.

27. The apparatus of claim 26 wherein the VIM comprises a plurality of VIM lines, and the means for determining an interference graph further comprises:

means for inserting an undirected edge into the interference graph between two VLIW nodes if the two VLIW instructions belong to a live-out set of the same node of the VLIW flow graph; and

means for coloring the interference graph nodes such that adjacent interference nodes are colored in different colors and each color corresponds to a different VIM line.

28. The apparatus of claim 19 wherein the lifetime of a VLIW instruction is a time interval extending from when said VLIW is defined by a load VLIW instruction to when said VLIW is last executed by an execute VLIW instruction.

29. The apparatus of claim 19 further comprising:

means for merging two non-overlapping VLIWs to share a common VIM line only when colorability of a resulting VLIW interference graph does not worsen as a result of said merging.

30. The apparatus of claim 19 further comprising:

means for utilizing a coalescing heuristic to reduce VIM requirements of a program.

31. The apparatus of claim 30 wherein said means for utilizing a coalescing heuristic produces a coalesced VIM address holding two or more of said plurality of VLIW instructions.

32. The apparatus of claim 19 further comprising:

means for shortening the life of a particular VLIW by placing an initialization LV statement adjacently prior to the use of its corresponding XV statement.

33. An apparatus for optimizing the execution time of a user program by reducing redundant loads of very long instruction word (VLIW) instruction memory (VIM) comprising:

means for selecting a load VIM (LV) instruction in a current node; and

means for placing the LV instruction in a new node which is closer to a program start node if an execution frequency of the new node is lower than an execution frequency of the current node, and if a maximum number of VIM lines is not exceeded.

34. An apparatus for statically determining liveness of indirect very long instruction word (VLIW) instructions comprising:

means for determining a control flow graph which includes nodes representing basic program blocks containing VLIW instructions, and edges connecting the nodes which represent jumps and calls from one block to another block; and

means for determining a live-in set and a live-out set of VLIW instructions for each node in the control graph to define a VLIW flow graph, a live-in set for a node comprises the VLIW instructions that are used in the node, a live-out set for a node comprises a union of live-in sets of successor nodes, the determining step further including solving VLIW flow equations for the live-in set and the live-out set.

35. The apparatus of claim 34 wherein the VLIW flow equations comprise:

$I_n = U_n \cup (O_n - D_n)$; and

$O_n = \cup_{s \text{ in succ}(n)} I_s$;

where "n" is a given node, I_n is a set of live-in VLIWs at node "n", O_n is a set of live-out VLIWs at node "n", U_n is a set of VLIWs that are used in "n", D_n is a set of VLIWs that are defined in "n", the live-out VLIWs of node "n" are all the VLIWs that belong to live-in sets of successor nodes of "n", and the notation $\cup_{s \text{ in succ}(n)} I_s$ denotes the union of all sets I_s where s is a successor node to node n.

36. An apparatus statically determining interference between indirect very long instruction word (VLIW) instructions from a control graph having a plurality of nodes, the apparatus comprising:

means for determining live-out sets for the plurality of nodes, the live-out sets and the control graph defining a VLIW flow graph;

means for determining an interference graph from the VLIW flow graph, the interference graph comprising VLIW nodes in which every VLIW node of the interference graph corresponds to one VLIW instruction;

means for inserting an undirected edge into the interference graph between two VLIW nodes if the two VLIW instructions belong to a live-out set of the same node of the VLIW flow graph; and

means for coloring the VLIW graph nodes such that adjacent VLIW nodes are colored in different colors and each color corresponds to a different VIM line.

EVIDENCE APPENDIX

None.

RELATED PROCEEDINGS APPENDIX

None.